

- `abs(Number)` Returns the absolute value of `value`, The "absolute value" of a number is its distance from zero as a positive number.
 - `acos(Number)` Returns the inverse cosine of `value`
 - `add_days(Number, Date)` Adds `number` days to `date`
 - `add_hours(Number, Date)` Adds `number` hours to `date`
 - `add_minutes(Number, Date)` Adds `number` minutes to `date`
 - `add_months(Number, Date)` Adds `number` months to `date`
 - `add_seconds (Number, Date)` Adds `number` seconds to `date`
 - `add_years(Number, Date)` Adds `number` years to `date`
 - `asin (Number)` Returns the inverse sine of `value`
 - `atan(Number)` Returns the inverse tangent of `value`
 - `beta_dist(Number, Number, Number, YesNo)` Returns the position of `value` on the beta distribution with parameters `alpha` and `beta`. If `cumulative = yes`, returns the cumulative probability
 - `beta_inv(Number, Number, Number)` Returns the position of `probability` on the inverse cumulative beta distribution with parameters `alpha` and `beta`
 - `binom_dist(Number, Number, Number, YesNo)` Returns the probability of getting `num_successes` successes in `num_tests` tests with the given `probability` of success. If `cumulative = yes`, returns the cumulative probability
-
- `binom_inv(Number, Number, Number)` Returns the smallest number `k` such that `binom(k, num_tests, test_probability, yes) >= target_probability`
 - `case(Any, Any,...)` Returns `value_if_yes` for the first `when` case whose `yesno_arg` value is `yes`. Returns `else_value` if all when cases are `no`.
 - `ceiling(Number)` Returns the smallest integer greater than or equal to `value`
 - `chisq_dist(Number, Number, YesNo)` Returns the position of `value` on the gamma distribution with `dof` degrees of freedom. If `cumulative = yes`, returns the cumulative probability
-
- `chisq_inv(Number, Number)` Returns the position of `probability` on the inverse cumulative gamma distribution with `dof` degrees of freedom
-
- `chisq_test(Number or Number List, Number or Number List chisq_test(actual, expected)`

- Returns the probability for the chi-squared test for independence between `actual` and `expected` data. `actual` can be a column or a column of lists, and `expected` must be the same type.
- `coalesce(Any, Any,...)` Returns the first non-`null` value in `value_1, value_2, ..., value_n` if found and `null` otherwise
- `combin(Number, Number)` Returns the number of ways of choosing `selection_size` elements from a set of size `set_size`
- `concat(Any, Any,...)` Returns `value_1, value_2, ..., value_n` joined as one string
- `confidence_norm(Number, Number, Number)` Returns half the width of the normal confidence interval at significance level `alpha`, standard deviation `stdev`, and sample size `n`
- `confidence_t(Number, Number, Number)` Returns half the width of the Student's t confidence interval at significance level `alpha`, standard deviation `stdev`, and sample size `n`
- `contains(String, String)` Returns `yes` if `string` contains `search_string`, and `No` otherwise
- `correl (Number, Number)` Returns the correlation coefficient of `column_1` and `column_2`
- `cos (Number)` Returns the cosine of `value`
- `count (Any)` Returns the count of non-`null` values in the column defined by `expression`, unless `expression` defines a column of Lists, in which case returns the count in each List
- `count_distinct(Any)` Returns the count of distinct non-`null` values in the column defined by `expression`, unless `expression` defines a column of Lists, in which case returns the count in each List
- `covar-pop(Number, Number)` Returns the population covariance of `column_1` and `column_2`

- `covar_samp(Number, Number)` Returns the sample covariance of `column_1` and `column_2`
- - `date(Number, Number, Number)` Returns "`year-month-day`" date or `null` if the date would be invalid
 - `date_time(Number, Number, Number, Number, Number, Number, Number)` Returns "`year-month-day hours:minutes:seconds`" date or `null` if the date would be invalid
- - `degrees(Number)` Converts `value` from radians to degrees
- - `diff_days(Date, Date)` Returns the number of days between `start_date` and `end_date`
- - `diff_hours(Date, Date)` Returns the number of hours between `start_date` and `end_date`
- - `diff_minutes(Date, Date)` Returns the number of minutes between `start_date` and `end_date`
- - `diff_months(Date, Date)` Returns the number of months between `start_date` and `end_date`
- - `diff_seconds(Date, Date)` Returns the number of seconds between `start_date` and `end_date`

- `diff_years(Date,Date)` Returns the number of years between `start_date` and `end_date`
- `exp(Number)` Returns e to the power of `value`
- `Expon_dist(Number,Number,YesNo)` Returns the position of `value` on the exponential distribution with parameter `lambda`. If `cumulative = yes`, returns the cumulative probability
- `Extract_days(Date)` Extracts the days from `date`
- `Extract_hours(Date)` Extracts the hours from `date`
- `Extract_minutes(Date)` Extracts the minutes from `date`
- `Extract_months(Date)` Extracts the months from `date`
- `Extract_seconds(Date)` Extracts the seconds from `date`

- `Extract_years(Date)` Extracts the years from `date`
- `f_dist(Number, Number, Number, YesNo)` Returns the position of `value` on the F distribution with parameters `dof_1` and `dof_2`. If `cumulative = yes`, returns the cumulative probability
- `f_inv(Number, Number, Number)` Returns the position of `probability` on the inverse cumulative F distribution with parameters `dof_1` and `dof_2`
- `fact (Number)` Returns the factorial of `value`
- `floor(Number)` Returns the largest integer less than or equal to `value`
- `gamma_dist(Number, Number, Number, YesNo)` Returns the position of `value` on the gamma distribution with parameters `alpha` and `beta`. If `cumulative = yes`, returns the cumulative probability
- `gamma_inv(Number, Number, Number)` Returns the position of `probability` on the inverse cumulative gamma distribution with parameters `alpha` and `beta`
- `geomean(Number or Number List)` Returns the geometric mean of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the geometric mean of each List

- `hypgeom_dist(Number, Number, Number, Number, YesNo)` Returns the probability of getting `sample_successes` from the given `sample_size`, number of `population_successes`, and `population_size`. If `cumulative = yes`, returns the cumulative probability
- `if(YesNo, Any, Any)` If `yesno_expression` evaluates to `yes`, returns the `value_if_yes` value. Otherwise, returns the `value_if_no` value
- `index(Any, Number)` Returns the value of the `n`th element of the column created by `expression`, unless `expression` defines a column of Lists, in which case returns the `n`th element of each list
- `intercept(Number, Number)` Returns the intercept of the linear regression line through the points determined by `y_column` and `x_column`
- `is_null(Any)` Returns `yes` if `value` is `null`, and `No` otherwise
- `kurtosis(Number or Number List)` Returns the sample excess kurtosis of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the sample excess kurtosis of each List
- `Large(Number or Number List, Number)` Returns the `k`th largest value of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the `k`th largest value of each List

- Length(String) Returns the number of characters in `string`
- list(Any...) Creates a List out of the given values
- Ln(Number) Returns the natural logarithm of `value`
- Log(Number) Returns the base 10 logarithm of `value`
- Lookup(Any, Any, Any) Returns the value in `result_column` that is in the same row as `value` is in `lookup_column`
- Lower (String) Returns `string` with all characters converted to lower case
- match(Any, Any) Returns the row number of the first occurrence of `value` in the column created by `expression` unless `expression` defines a column of Lists, in which case returns the position of `value` in each List
- max(Number or Number List) Returns the max of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the max of each List
- mean(Number or Number List) Returns the mean of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the mean of each List

- `median(Number or Number List)` Returns the median of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the median of each List
- `min(Number or Number List)` Returns the min of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the min of each List
- `mod(Number, Number)` Returns the remainder of dividing `value` by `divisor`
- `mode(Number or Number List)` Returns the mode of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the mode of each List
- `multinomial(Number, ...)` Returns the factorial of the sum of the arguments divided by the product of each of their factorials
- `negbinom_dist(Number, Number, Number, YesNo)` Returns the probability of getting `num_failures` failures before getting `num_successes` successes, with the given `probability` of success. If `cumulative = yes`, returns the cumulative probability
- `norm_dist(Number, Number, Number, YesNo)` Returns the position of `value` on the normal distribution with the given `mean` and `stdev`. If `cumulative = yes`, then returns the cumulative probability

- `norm_inv(Number, Number, Number)` Returns the position of `probability` on the inverse normal cumulative distribution
- `norm_s_dist(Number, YesNo)` `norm_s_inv(Number)` Returns the position of `value` on the standard normal distribution. If `cumulative = yes`, returns the cumulative probability
- `NOT` Returns `yes` if `value` is `No`, and `yes` otherwise
- `now()` Returns the current date and time
- `offset(Any, Number)` Returns the value of row `(n + row_offset)` in `column`, where `n` is the current row number
- `offset_list(Any, Number, Number)` Returns a List of the `num_values` values starting at row `(n + row_offset)` in `column`, where `n` is the current row number
- `percent_rank(Number or Number List, Number)` Returns the rank of `value` in `column` as a percentage from 0 to 1 inclusive
- `percentile(Number or Number List, Number)` Returns the value from the column created by `expression` corresponding to the given `percentile_value`, unless `expression` defines a column of Lists, in which case returns the percentile value for each List. `percentile_value` must be between 0 and 1, else this returns `null`

- `pivot_column()` Returns the index of the current pivot column
- `pivot_index(Any, Number)` Evaluates `expression` in the context of the pivot column at position `pivot_index` (1 for first pivot, 2 second pivot, etc.). Returns null for unpivoted results
- `pivot_offset(Any, Number)` Returns the value of the `pivot_expression` in position (`n + column_offset`), where `n` is the current pivot column position. Returns null for unpivoted results
- `pivot_offset_list(Any, Number, Number)` Returns a List of the `num_values` values in `pivot_expression` starting at position (`n + column_offset`), where `n` is the current pivot index. Returns null for unpivoted results
- `pivot_row(Any)` Returns the pivoted values of `expression` as a List. Returns null for unpivoted results.
- `pivot_where(YesNo, Any)` Returns the value of `expression` for the pivot column which uniquely satisfies `select_expression` or null if such a column does not exist or is not unique.

- `poisson._dist(Number, Number, YesNo)` Returns the position of `value` on the poisson distribution with parameter `lambda`. If `cumulative = yes`, returns the cumulative probability
- `position(String, String)` Returns the start index of `search_string` in `string` if it exists, and `0` otherwise
- `power(Number, Number)` Returns `base` raised to the power of `exponent`
- `product(Number or Number List)` Returns the product of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the product of each List
- `radians(Number)` Converts `value` from degrees to radians
- `rand()` Returns a random number between 0 and 1
- `rank(Number, Number or Number List)` Returns the rank of `value` in the column created by `expression` unless `expression` defines a column of Lists, in which case returns the rank of `value` in each List
- `rank._avg(Number, Number or Number List)` Returns the average rank of `value` in the column created by `expression` unless `expression` defines a column of Lists, in which case returns the average rank of `value` in each List

- `replace(String, String, String)` Returns `string` with all occurrences of `old_string` replaced with `new_string`
- `round(Number, Number)` Returns `value` rounded to `num_decimals` decimal places
- `running_product(Number)` Returns a running product of the values in `value_column`
- `running_total (Number)` Returns a running total of the values in `value_column`
- `sin(Number)` Returns the sine of `value`
- `skew(Number or Number List)` Returns the sample skewness of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the sample skewness of each List
- `slope(Number, Number)` Returns the slope of the linear regression line through points determined by `y_column` and `x_column`
- `small(Number or Number List, Number)` Returns the `kth` smallest value of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the `kth` smallest value of each List
- `split(String, String)` Returns a `List` of strings in `string` broken up by `delimiter`

- `sqrt(Number)` Returns the square root of `value`
- `stddev_pop(Number or Number List)` Returns the standard deviation (population) of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the standard deviation (population) of each List
- `stddev_samp(Number or Number List)` Returns the standard deviation (sample) of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the standard deviation (sample) of each List
- `substring(String, Number, Number)` Returns the substring of `string` beginning at `start_position` consisting of `length` characters
- `sum(Number or Number List)` Returns the sum of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the sum of each List
- `t_dist(Number, Number, YesNo)` Returns the position of `value` on the student's t-distribution with `dof` degrees of freedom. If `cumulative = yes`, returns the cumulative probability
- `t_inv(Number, Number)` Returns the position of `probability` on the inverse normal cumulative distribution with `dof` degrees of freedom

- `t_test(Number, Number, Number, Number)` Returns the result of a Student's t-test on the data from `column_1` and `column_2`, using 1 or 2 tails. `type`: 1 = paired, 2 = homoscedastic, 3 = heteroscedastic
- `tan(Number)` Returns the tangent of `value`
- `to_date(String)` Returns the date and time corresponding to `string` (YYYY, YYYY-MM, YYYY-MM-DD, YYYY-MM-DD hh, YYYY-MM-DD hh:mm, or YYYY-MM-DD hh:mm:ss)
- `to_number (String)` Returns the number represented by `string`, or `null` if the string cannot be converted
- `to_string(Any)` Returns the string representation of `value`, or an empty string if `value` is null
- `trunc_days(Date)` Truncates `date` to days
- `trunc_hours(Date)` Truncates `date` to hours
- `trunc minutes(Date)` Truncates `date` to minutes
- `trunc_months(Date)` Truncates `date` to months

- `trunc_years(Date)` Truncates `date` to years
- `upper(String)` Returns `string` with all characters converted to upper case
- `var_pop(Number or Number List)` Returns the variance (population) of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the variance (population) of each List
- `var_samp(Number or Number List)` Returns the variance (sample) of the column created by `expression` unless `expression` defines a column of Lists, in which case returns the variance (sample) of each List
- `weibull_dist(Number, Number, Number, YesNo)` Returns the position of `value` on the Weibull distribution with parameters `shape` and `scale`. If `cumulative = yes`, returns the cumulative probability
- `when(YesNo, Any)` Returns `value_if_yes` if `yesno_expression` evaluates to `yes`. Otherwise returns `null`
- `Z_test(Number, Number, Number)` Returns the one-tailed p-value of the z-test using the existing `data` and `stdev` on the hypothesized mean `value`.

